Proceedings of the ASME 2015 International Technical Conference and Exhibition on Packaging and Integration of Electronic and Photonic Microsystems and ASME 2015 12th International Conference on Nanochannels, Microchannels, and Minichannels
InterPACKICNMM2015
July 6-9, 2015, San Francisco, USA

InterPACKICNMM2015-48568

EXPERIMENTAL VALIDATION OF A DETAILED PHASE CHANGE MODEL ON A HARDWARE TESTBED

Charlie De Vivero
Department of Electrical and Computer Engineering
Boston University
Boston, MA 02215
Email: devivero@bu.edu

Fulya Kaplan
Department of Electrical and Computer Engineering
Boston University
Boston, MA 02215
Email: fkaplan3@bu.edu

Ayse K. Coskun
Department of Electrical and Computer Engineering
Boston University
Boston, MA 02215
Email: acoskun@bu.edu

ABSTRACT

Continued CMOS scaling accompanied with a stall in the voltage scaling has led to high on-chip power densities. High on-chip power densities elevate the temperatures, substantially limiting the performance and reliability of computing systems. The use of Phase Change Materials (PCMs)[1] *has been explored as a passive cooling method to manage excessive chip temperatures. The thermal properties of PCMs allow a large amount of heat to be stored at near-constant temperature during the phase transition. This heat storage capability of PCM can be leveraged during periods of intense computation. For systems with PCM, development of new management strategies is essential to maximize the benefits of PCM. In order to design and evaluate these management strategies, it is necessary to have an accurate PCM thermal model. In our recent work, we proposed a detailed phase change thermal model, which we integrated into a compact thermal simulation tool, HotSpot. In this paper, we build a hardware testbed incorporating a PCM unit on top of the chip package. We then validate the accuracy of our previously proposed thermal model by comparing the HotSpot simulation results against the measurements on the testbed. We observe that the error between the measured and simulated temperatures is less than* 4^oC *with* 0.65 *probability. Finally, we implement a soft PCM capacity sensor that monitors the remaining PCM latent heat capacity to be used for development of thermal management policies. We evaluate a set of thermal management policies on the testbed. We compare policies that adjust the sprinting frequency based on current temperature against the policies that take action based on the remaining PCM capacity.*

NOMENCLATURE

PCM	Phase Change Material
V/f	Voltage/frequency
T	Temperature
T_1	Onset temperature of phase change
T_2	End temperature of phase change
c_{ps}	Specific heat capacity of the PCM in solid phase
c_{tr}	Specific heat capacity of the PCM during phase transition
c_{pl}	Specific heat capacity of the PCM in liquid phase
T_{CPU}	Average temperature of the CPU cores
T_{PCM}	PCM temperature
T_{AMB}	Ambient temperature
P_{NET}	Net power input to the PCM
$E_{STORED,t}$	Latent heat energy stored in the PCM at time t
$R_{Si\_to\_PCM}$	Equivalent thermal resistance from silicon to PCM
$R_{PCM\_to\_AIR}$	Equivalent thermal resistance from PCM to air
$t_{sampling}$	Temperature sampling interval
c_i	Fitting constants for equivalent resistance calculation

[1] PCM has also been used when referring to phase change memory in the literature. This paper focuses on using phase change materials as thermal buffers, and not on memories built with PCM.

INTRODUCTION

Thermal management of processors is becoming an increasingly difficult challenge, as CMOS scaling trends lead to higher on-chip power density by allowing more transistors in a smaller area. High temperatures are especially challenging in the mobile space where devices can typically only accommodate passive cooling methods (i.e., active cooling methods require additional power, area, and cost, and thus are often not feasible). Elevated temperatures increase leakage power and also degrade performance, as the built-in mechanisms on the processors turn off or slow down cores in case of a thermal emergency.

To address the challenge of implementing thermal efficiency without using active cooling, Phase Change Materials (PCMs) have been used as a passive cooling solution [1–4]. PCM can store large amounts of heat, referred to as *latent heat*, during phase change at a close-to-constant temperature. Having this thermal property, PCM acts as a large thermal capacitor, limiting the rise in temperature during phase transition. Thus, it provides a smoother chip thermal profile and allows performance boosting strategies such as *computational sprinting* [1].

In *computational sprinting* the system temporarily exceeds its Thermal Design Power (TDP) to respond to short bursts of intense computation. In this context, PCM has been used to extend sprinting duration for higher performance gains [1–4]. For the design and realistic evaluation of management strategies leveraging PCM, having an accurate PCM model is essential. Failing to do so results in inefficient use of the PCM capabilities and suboptimal management policies. Recent work shows the negative impact of using inaccurate models on the resulting runtime management decisions [5].

A significant portion of the existing work on PCM-oriented thermal management relies on simpler phase change models [1, 3]. Other work in the area demonstrates the benefits of PCM on a thermal test chip [4], or on a testbed [2]. Having a testbed has many advantages in terms of showing the applicability of the proposed strategies and including the real-life effects that are hard to capture in a simulation environment. On the other hand, simulation with an accurate PCM model is necessary to design policies that target a variety of platforms with different technology nodes, core counts, power consumption levels and such. For this purpose, in our prior work we proposed a detailed PCM thermal model [5], and validated the accuracy of the model by comparing its results against computational fluid dynamics (CFD) simulations. However, none of the previously proposed PCM models has been compared against real-life measurements on a testbed with PCM.

Our specific contributions in this paper are as follows:

- We build a hardware testbed with a PCM unit installed on top of the chip package. We create a model of our testbed using HotSpot [6] thermal simulator with the integrated detailed PCM model from our previous work [5]. We then validate the accuracy of the PCM model by comparing the temperature traces obtained from HotSpot simulations against the ones measured on the testbed. Transient simulations show that the error between the measured and simulated temperatures is less than 4^oC with 0.63 probability.

- We design a soft PCM capacity sensor that monitors the remaining PCM capacity (i.e., the remaining amount of latent heat energy that can be stored in the PCM) to be used as part of practical PCM-aware runtime management policies. Having a soft PCM sensor is beneficial as it gives information on the PCM state and how much longer the system can *sprint*.

- We evaluate a set of thermal management policies on the testbed. We compare policies that adjust the sprinting frequency based on current temperature against the policies that take action based on the remaining PCM capacity. Our results show that proactively monitoring the PCM capacity improves performance by up to 4.5% compared to the baseline DVFS policy.

RELATED WORK

PCM has been widely used in thermal management of processors. The existing work on PCM can be divided into two main groups: (1) using PCM as a heat spreader or heat sink enhancer, (2) exploiting PCM as part of performance boosting strategies.

The first group of work focuses on designing more efficient heat spreader or heat sink units by incorporating PCM in the cooling package [7–11]. Tan et al. show the benefit of PCM by performing CFD simulations on a mobile phone with a PCM filled heat storage unit [8]. Alawadhi et al. study the effectiveness of a thermal control unit involving PCM and a thermal conductivity enhancer on a portable electronic device using experimental and numerical analysis [7]. Other research aim at saving energy by designing hybrid heat sinks incorporating PCM [9,10]. Low thermal conductivity of the PCM is a major limiter on its benefits. Recent work addresses this problem by proposing the use of metal-PCM composites as heat spreaders in mobile devices [11]. In this work, authors show the tradeoff between thermal conductivity and latent heat capacity by performing a parametric analysis on the metal fraction of the composite.

The second group of work centers around designing performance boosting policies that exploit the PCM properties [1–4]. Raghavan et al. introduce *computational sprinting*, exceeding the TDP of the processor temporarily to respond to short durations of intense computation. PCM has been proposed to extend sprinting duration in this context and a simple PCM model has been used to simulate the benefits of PCM. The follow up of this work verifies the feasibility of *computational sprinting* on a hardware/software testbed [2]. The concept of *sprint pacing* is intro-

duced in this work, which is switching to lower intensity sprinting mode when half of the PCM has melted. Other research in the area focuses on developing techniques to sprint periodically for longer durations [3, 4]. Tilli et al. propose *computational re-sprinting* for periodic hard deadline tasks, which adjusts the voltage/frequency (V/f) settings of the cores to reserve the minimum amount of PCM capacity for the next sprinting period [3]. They evaluate the benefits of their policy using a simple PCM model and simulations. Shao et al. consider repeated sprints by alternating between *sprint* and *rest* modes based on a predefined duty cycle [4]. They evaluate the benefits of the technique on a thermal test chip with an integrated on-chip phase change heat sink as a proxy for a smart phone processor. Kaplan et al. propose a detailed PCM thermal model [5], and validate it against a CFD model.

Some of the prior work on PCM management monitors the PCM state (i.e., the remaining PCM capacity) as part of the sprinting strategy [1, 3]. However, they implement the PCM monitors in simulation environment based on a priori characterization of power and temperature. For example, Raghavan et al. use McPat to estimate the energy consumption of the cores and assume that all of this energy is stored in the PCM [1]. On the other hand, Tilli et al. use a simpler PCM thermal model, where the PCM temperature is assumed to stay constant at all points on the PCM during melting [3].

Our work is the first to experimentally validate a PCM thermal model on a hardware testbed. Our testbed uses a mobile development platform and a custom-designed PCM unit on top. We run real benchmarks on the testbed and show the accuracy of our PCM thermal model [5]. We also implement and evaluate for the first time a soft PCM capacity sensor that monitors the remaining PCM capacity at runtime based temperature measurements and equivalent thermal resistances of the package on a hardware testbed.

METHODOLOGY

Testbed Setup

We design and build the hardware testbed with the following goals in mind:

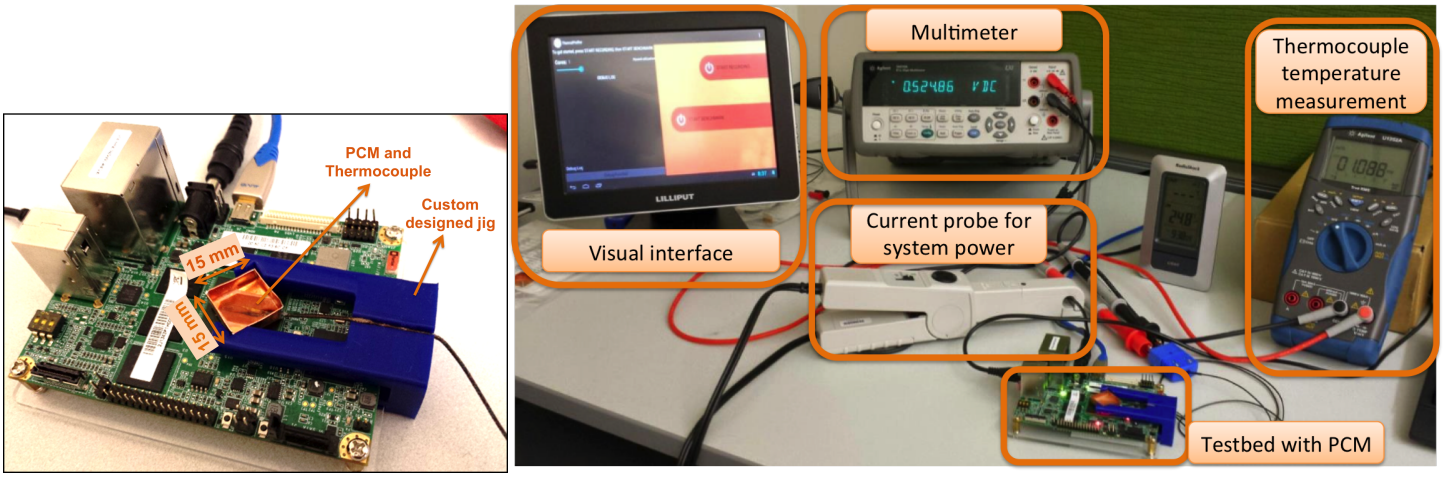
- Provide a computing platform that acts as a proxy for a mobile computing device fitted with a PCM-based thermal management solution.
- Provide data acquisition setup that measures the CPU power consumption, CPU core temperatures, and PCM temperature in order to evaluate the performance of the PCM-based thermal management solution.

Computing Platform. We use an Inforce Computing IFC6410 single-board computer (SBC) as our computing platform. The platform is powered by a Qualcomm Snapdragon 600 System-on-Chip (SoC), which includes a quad-core 1.2 GHz mobile processor (with a 2 MB shared L2 cache) commonly found in modern mobile devices. The IFC6410 provides 2.0 GB of RAM, and runs Android 4.1. The Snapdragon processor does not have a heat sink, thus, the processor is normally exposed to ambient air. We build a copper box enclosure that holds the PCM, and fit the copper box enclosure with a thermocouple that is directly exposed to the PCM. The physical dimensions of the copper box are $15mm \times 15mm \times 5.5mm$. We place the PCM enclosure on top of the Snapdragon processor and hold them together with compressive force using a custom-designed jig. We use a single thermocouple to measure the PCM temperature and it is placed at the bottom surface of the copper PCM container. Thermal Interface Material (TIM), which is made of *Arctic Silver 5* compound, lies in between the processor die and the PCM enclosure. Figure 1(a) shows the PCM enclosure mounted on the SBC. The processor is located directly under the copper box. We use paraffin wax as the phase change material. We use $0.175g$ of PCM, which corresponds to a $1.02mm$ thickness. We report the thermal specifications of the TIM and PCM in Table 2.

Data Acquisition. The data acquisition equipment records the total power consumption of the SBC and the thermocouple temperature, which is interpreted as the temperature of the PCM layer. The Snapdragon SoC provides internal temperature sensors which give readings on the individual CPU core temperatures. The SBC runs our custom-built Android application to read and record the CPU core temperature sensors, provides a means for running benchmark tests, and employs thermal management policies.

Figure 1(b) illustrates our measurement setup. Power measurements are sampled at a rate of 70 Hz using the Agilent 34410A multimeter in conjunction with the Agilent 34134A current probe. PCM layer temperature is measured via the thermocouple and recorded by the Android application at a rate of 1.0 Hz. CPU core temperatures are recorded by the Android application at a rate of 1.0 Hz. The CPU utilization incurred by the Android application while recording these temperatures is less than 0.1%.

Benchmark Applications. We run a selection of computational kernels from the SciMark 2.0 Java benchmark [12], with small problem sizes to focus on exercising CPU-intensive loads on the testbed, i.e., the dataset used by the application is completely contained in the cache. The selected applications are (i) *Jacobi Successive Over-Relaxation (sor)*, (ii) *Sparse Matrix Multiply (smult)*, and (iii) *Dense LU Matrix Factorization (lu)*. *Sor* exercises typical access patterns in finite difference methods, *smult* applies matrix multiplication on an unstructured sparse matrix, *lu* computes the LU factorization of a dense 100×100 matrix.



(a) (b)

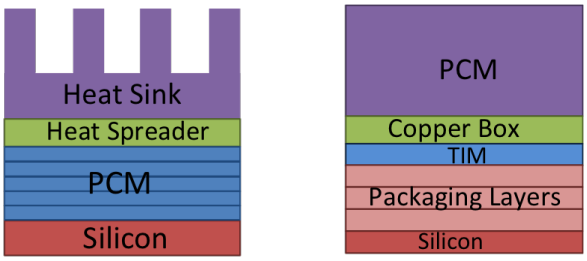
FIGURE 1: (a) IFC6410 SBC with copper box holding PCM, fitted on top of the Qualcomm Snapdragon SoC, (b) Experimental setup including the computing platform and the data acquisition equipment.

Modeling Phase Change in Thermal Simulators

We use the detailed phase change model proposed in our recent work [5], which was validated against a CFD model and provided 0.27^oC root mean square error. This model was integrated into a compact thermal simulator, HotSpot [6]. The PCM model uses the basic 3D stacking feature in HotSpot, allowing the user to define multiple layers of any desired material. Grid-level simulation granularity provides fine-grained simulation by dividing the floorplan into small cells and computing the temperature for each grid cell.

Our PCM thermal model uses the *apparent heat capacity* method [7] to model phase change behavior from solid to liquid or vice versa. In this method, a nonlinear temperature-dependent specific heat capacity is assigned to the PCM. Phase change from solid to liquid occurs over a temperature interval, during which the specific heat capacity of the PCM is set to a very high value compared to the one in solid and liquid phases. A very high specific heat capacity during phase transition indicates a very low rate of change of temperature, and thus mimics the close-to-constant temperature behavior of melting. The integral of the specific heat capacity over the temperature interval represents the *latent heat of fusion* stored in the PCM.

In our recent work [5] we modeled an on-chip PCM that lies between the silicon die and the heat spreader. To implement the model in HotSpot, we inserted a layer of PCM that lies on top of the silicon layer as shown in Figure 2(a). The PCM layer has the same floorplan as the silicon chip but it does not dissipate any power. We modified HotSpot to define the melting point and latent heat of fusion of the PCM. Each PCM grid cell is assigned a temperature-dependent specific heat capacity as in Equation (1). The specific heat capacity of each PCM grid cell is updated at every time step following Equation (1):



(a) (b)

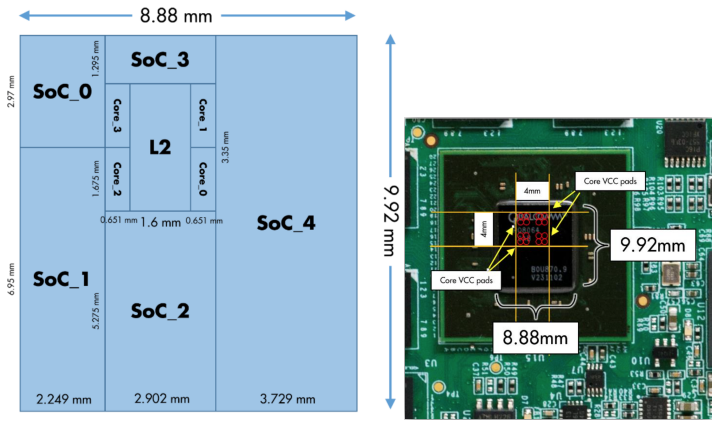
FIGURE 2: (a) Example chip package incorporating PCM; (b) Snapdragon chip package as we modeled in HotSpot.

$$C_{p,pcm}(T) = \begin{cases} c_{ps} & T < T_1 \\ c_{tr} & T_1 \leq T \leq T_2 \\ c_{pl} & T > T_2 \end{cases} \quad (1)$$

where c_{ps}, c_{pl}, and c_{tr} are the specific heat capacities of solid, liquid, and phase transition states, respectively. We use $c_{ps} = c_{pl}$ as in prior work [13]. T_1 is the onset temperature and T_2 is the end temperature of the phase transition. In our simulations, we use a transition temperature interval of 1^oC. We use hexacosane paraffin as our PCM compound [14]. The melting temperature of the paraffin PCM is 55^oC, which corresponds to the center point of the (T_1, T_2) interval. We set $c_{ps} = 1.41 \cdot 10^6 J/m^3K$, and $c_{tr} = 190.5 \cdot 10^6 J/m^3K$ based on the physical properties of hexacosane.

Modeling the Snapdragon Platform in HotSpot

For experimental validation of the proposed PCM thermal model, we first need to create a physical model of the Snapdragon platform in HotSpot. This is a rather challenging process because



(a) (b)

FIGURE 3: (a) Snapdragon SoC floorplan used in the HotSpot model; (b) Inferred location of the CPU cores shown on the chip.

we have limited information on the layout, specific locations of the units and the power consumptions of the CPU cores. Thus, it is essential to have good estimates on the geometry and the physical/thermal properties of the platform, as they have a significant impact on the simulation results. This section describes how we address this modeling challenge.

Snapdragon Floorplan. A detailed floorplan of the Qualcomm Snapdragon 600 SoC that shows the locations and the sizes of the cores and other units is not available. In order to estimate the area of the individual CPU cores and the shared L2 cache, we use McPat [15], an architectural modeling tool for multicore processors. The total estimated area is $1.09mm^2$ for a CPU core and $5.36mm^2$ for the L2 cache. The specific locations of the CPU cores on the floorplan are estimated based on the position of the dedicated core power supply (VCC) pads on the underside of the package as seen in Figure 3(b). We define the rest of the SoC area (i.e., the un-core area) as rectangular blocks. Figure 3(a) shows the Snapdragon SoC floorplan we used in the HotSpot model.

Snapdragon Chip Layer Stack. We measure the thickness of the Snapdragon chip as $900um$, which includes the silicon die and some packaging material. We estimate the silicon thickness as $100um$, which is typical at the 28nm technology node. For the remaining $800um$, we define thin layers (each $200um$-thick) of packaging material (See Figure 2(b)). Information on the packaging material and its properties are not available. The general approach in electronic packaging today is to use materials that provide both electrical insulation and good thermal conduction such as mica, silicon-impregnated cloth, and ceramic materials. Examples include silicon dioxide (SiO_2) and silicon nitride (Si_3N_4) ceramic. Si_3N_4 is used as a final passivation layer on chips. It prevents underlying silicon from being oxidized. Thus, we assume Si_3N_4 in this work [16, 17].

As explained in the *Testbed Setup* section, we place a

Board components	System PMIC	27.083 mW
	DDR3 PMIC	3.0149 mW
	DDR3 (Standby Power Consumption)	258 mW
On-chip L2 cache	Standby power consumption	344 mW
	Peak power consumption	847 mW
Other on-chip components		1.04W – 1.31W
CPU power (split among active cores)		0.1 W – 11.9 W

TABLE 1: Power values of the components on the board.

$15mm \times 15mm$ copper box on top of the processor to contain the PCM. We modify the *heat spreader* properties in HotSpot to account for the bottom surface of the copper box. As the Snapdragon SoC does not have a heat sink while HotSpot software does not allow the removal of the heat sink, we model the *heat sink* as our PCM by integrating the phase change model into the package. We use a $1.02mm$-thick paraffin wax as the PCM and modify the heat sink parameters accordingly. We assume the PCM does not vaporize and thus, the thickness of the PCM does not change over time. A thin TIM layer (made of Arctic Silver 5 [18]) lies between the copper box and the packaging layers to provide better contact. Figure 2(b) illustrates the Snapdragon chip layer stack as we modeled in HotSpot. Table 2 provides a summary of the material properties used in the HotSpot model.

Snapdragon Core and Un-core Power. On our testbed, we can measure the system power (i.e., the total power going into the SBC including CPU, RAM, and other chips on the board) as described in *Testbed Setup* section. However, in HotSpot we focus on the CPU, thus, we need the power consumption values of the individual CPU cores and the un-core units defined as SoC units in Figure 3(a). For this purpose, we first estimate the CPU power as some portion of the total power, based on the datasheets of the chips on the board. Note that we turn off wifi, GPS, and bluetooth units to isolate the CPU power. We do not use the GPU unit either. Knowing the CPU power, we find the un-core power based on the power difference observed when different number of cores are activated (e.g., difference between the cases when four cores are activated and zero cores are activated). We repeat this core power estimation process across different V/f settings and observe consistent trends. We distribute the un-core power to individual SoC units proportional to their areas. Table 1 reports the power values of the components on the board.

EXPERIMENTAL VALIDATION OF THE PHASE CHANGE MODEL

In this section, we explain the details of how we experimentally validate the proposed PCM thermal model on our hardware testbed. We carry out two main sets of comparisons: steady state temperature and transient temperature, and we report the errors for both cases.

Steady State Temperature Comparison

Steady state temperature is the temperature that the components reach after operating at a stable power for longer than the system's thermal time constant. For the steady state experiments, we consider the following configurations:

- We experiment with 4 different CPU frequency settings: 594, 810, 1026, or 1242 MHz.
- We turn on 1, 2, 3, or 4 CPU cores.
- We experiment with 4 different power traces: power traces of 3 benchmark applications (successive over-relaxation (*sor*), sparse matrix multiplication (*smult*), and lu factorization (*lu*)) and power at idle state. For the experiments at idle state, we turn on the cores, but keep them idle (i.e., no benchmark is running). For the experiments with benchmark applications, we run the applications until the system reaches a steady temperature. As the original running time of the applications are much shorter than the thermal time constant of the processor, we run them in loops to generate longer application traces.

During the experiments with a higher number of active cores and higher V/f levels, the temperatures rise to critical levels, thus, the built-in mechanisms throttle down the cores automatically. We eliminate those cases from our evaluations as the temperature never reaches a stable value (i.e., the power and temperature traces fluctuate continuously). By combining the configurations listed above and eliminating the unstable cases, we experiment with a total of 53 cases. For comparison purposes, we focus on the temperatures of the individual CPU cores and the thermocouple. The HotSpot counterpart of the thermocouple temperature is the average PCM temperature.

Steady State Calibration. The default thermal parameters in HotSpot represent an example processor package, which does not necessarily match the characteristics of our hardware testbed. Moreover, we do not have exact knowledge on the physical/thermal properties of each material on the Snapdragon processor. Thus, we start off with the material properties that we know of and we go through a calibration phase to match the thermal behaviors of the model and the testbed. Two main package parameters affect the steady state temperature: heat sink convection resistance and heat sink thermal conductivity. As the heat sink corresponds to the PCM, we initially assign the paraffin wax thermal conductivity (0.25 W/mK) to it. However, this is a pessimistic assumption and results in unreasonably high steady state temperatures in HotSpot. This is because in reality, the side surfaces of the copper box provides additional heat conduction improving the low thermal conductivity of the paraffin wax. We calibrate the PCM thermal conductivity to account for this effect and set it to 0.50 W/mK. The details of the package material properties are presented in Table 2 with calibrated values specified in parenthesis.

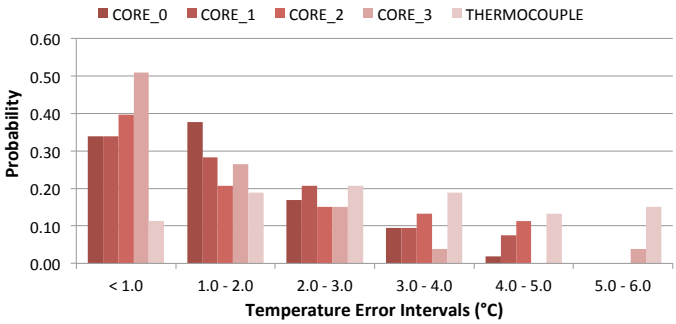


FIGURE 4: HotSpot model steady state error probability histogram.

Next, we compare the steady state temperatures measured on the testbed against the ones obtained through HotSpot simulations. Figure 4 is a histogram plot that shows the steady state error probability. The x-axis represents the temperature error intervals in ^oC and the y-axis shows the probability of having a temperature error within the corresponding interval. We generate this plot using the following approach: (1) For each steady state experiment, we compare the measured and simulated temperatures across all units, and find the absolute steady state temperature error for each unit. (2) To find the probability of having a temperature error less than 1^oC, for example, we count the number of times we encounter an error that is less than 1^oC. We then divide this number by the total number of steady state experiments. Error probability represents the degree of matching between our simulations using the phase change model and the real-life measurements. According to Figure 4, the steady state temperature error is less than 4^oC with 0.89 probability and less than 2^oC with 0.6 probability. The maximum, average and RMS error for steady state experiments is 6^oC, 1.9^oC, and 2.37^oC, respectively.

Transient Temperature Comparison

For the transient temperature comparison, we use the same set of configurations as described in the previous section, but this time we use the transient benchmark power traces as input. Each transient temperature experiment starts with an idle period of 10 seconds to let the system reach a stable temperature. We then run the benchmark for its corresponding number of loops. Finally we switch the system back to idle mode for a period of time. We adjust the duration of the second idle period such that each transient experiment corresponds to a 60-second time frame. In HotSpot, we use a 10ms sampling interval and initialize the simulations with their corresponding steady state temperatures.

Transient Calibration. Both the thermal resistance and the capacitance play a role in the transient behavior. We have already

Chip Layers	Density (g/cm^3)	Thickness (mm)	Length (mm) × Width (mm)	Thermal Conductivity (W/mK)	Specific Heat Capacitance (J/m^3K)	Convection Resistance (K/W)	Convection Capacitance (J/K)
Silicon	2.33	0.10	9.92 × 8.88	100.0	1.750×10^6	-	-
Packaging (Si_3N_4)	2.37	0.80	9.92 × 8.88	30.0	1.595×10^6	-	-
TIM (Arctic Silver 5)	-	0.05	9.92 × 8.88	8.7	1.750×10^6	-	-
Heat Spreader (Copper box)	8.96	0.25	15.00 × 15.00	400.0	3.450×10^6	-	-
Heat Sink (PCM) (Hexacosane)	0.762	1.02	15.00 × 15.00	0.25 (original) 0.50 (calibrated)	$c_{ps} = 1.41 \times 10^6$ $c_{tr} = 190.5 \times 10^6$	0.45 (calibrated)	20.0 (calibrated)

TABLE 2: HotSpot package material properties.

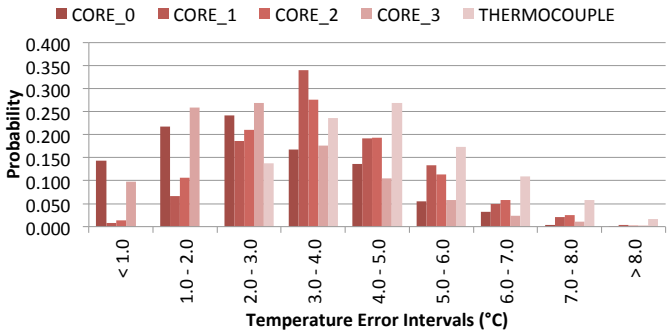


FIGURE 5: HotSpot model transient error probability histogram.

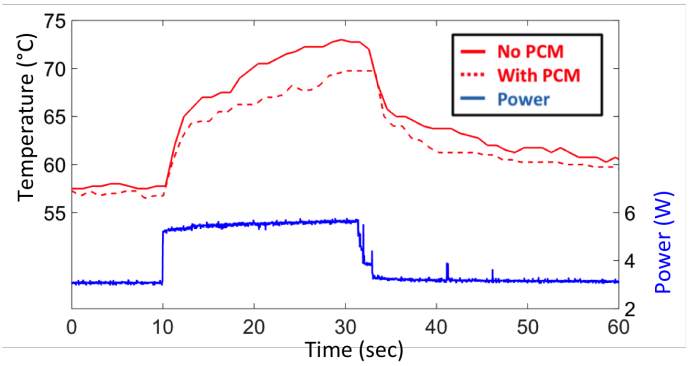


FIGURE 6: Comparison of temperatures for systems with and without PCM.

calibrated the thermal resistances for the steady state. Thus, we use the heat sink convection capacitance for transient calibration, which determines how fast the heat sink responds to a change in temperature. Table 2 shows the convection capacitance calibrated for our HotSpot testbed model.

Figure 5 represents the transient temperature error probability on a histogram plot. We follow a similar approach to generate this plot: (1) For each time step of the transient simulation, we compare the measured and simulated temperatures across all units, and find the absolute transient error for each unit. (2) To find the probability of having a temperature error less than 1^oC, for example, we count the number of times we encounter an error that is less than 1^oC. We then divide this number by the total number of simulation time steps. Figure 5 shows that the transient temperature error between the measured and simulated temperature is less than 4^oC with 0.63 probability. The temperature range in our experiments is 68^oC, where 4^oC of error corresponds to only 5.8%. On the other hand, the probability of having a temperature error that is higher than 8^oC is as low as 0.005. We also observe average and RMS error of 3.54^oC and 3.76^oC, respectively, for the transient simulations.

IMPACT OF PCM ON THERMAL PROFILES

Before exploring the management strategies leveraging PCM, we experimentally demonstrate the thermal benefits of using PCM as opposed to having no PCM.

Figure 6 compares the temperature traces measured on the testbed with and without PCM. Note that the experiments without PCM include the empty copper box. We run the *smult* application using all 4 cores at 810 MHz for both cases. Figure 6 shows that using PCM reduces the peak temperature by 4.5^oC. We carry out the same comparison for the other benchmarks, V/f levels and number of active cores. Table 3 reports the peak temperature reduction values and shows that using PCM reduces the peak temperature by up to 5.26^oC, and by 2.75^oC on average.

The benefit of PCM is not limited to peak temperature reduction. In the following sections, we demonstrate the performance benefits of leveraging PCM in cooperation with *computational sprinting* policies.

PCM CAPACITY SENSOR

Monitoring PCM capacity enables estimation of the remaining sprinting capability. We implement a soft PCM capacity sensor that monitors how much of the PCM remains unmelted at runtime. Our soft sensor targets real life use as part of thermal

	Across all experiments		Experiments with number of active cores:							
			1 core		2 cores		3 cores		4 cores	
	°C	%	°C	%	°C	%	°C	%	°C	%
Average	**2.75**	**4.1**	**1.03**	**1.9**	**2.38**	**3.8**	**3.56**	**5.2**	**4.03**	**5.4**
Maximum	5.26	7.6	1.25	2.5	5.25	7.8	4.75	7.2	5.26	7.6
Minimum	0.50	1.0	0.50	1.0	0.75	1.4	2.25	3.3	2.50	2.8
Std. Dev.	1.53	2.0	0.26	0.5	1.37	2.0	0.93	1.2	1.05	1.7

TABLE 3: Peak temperature reduction when using PCM compared to not using PCM. The table provides ^oC reduction and corresponding percentage.

management strategies. We then demonstrate a use case for the soft PCM sensor by utilizing it in cooperation with a PCM-aware thermal management policy.

Implementation of the PCM Sensor

The PCM capacity sensor is a counter that accumulates the amount of latent heat energy stored in the PCM at a given time, during phase change. At the beginning of the phase transition, the amount of latent energy stored in the PCM is zero. In order to fully melt, the PCM needs to store energy that is equal to its latent heat of fusion. The PCM sensor estimates this stored energy by using the temperature sensor measurements and thermal resistances of the package as follows: we estimate the heat entering the PCM from the silicon layer and exiting the PCM to the ambient air. Using the temperature measurements on the CPU cores and the PCM, we can estimate the net power entering the PCM by using the following formula:

$$P_{NET} = \frac{T_{CPU} - T_{PCM}}{R_{Si\_to\_PCM}} - \frac{T_{PCM} - T_{AMB}}{R_{PCM\_to\_AIR}} \quad (2)$$

$$E_{STORED,t} = E_{STORED,(t-1)} + P_{NET} \times t_{sampling} \quad (3)$$

where T_{CPU}, T_{PCM} and T_{AMB} are the temperatures of the CPU cores (we use the average of the four CPU cores), the PCM, and ambient air, respectively. $R_{Si\_to\_PCM}$ and $R_{PCM\_to\_AIR}$ are the equivalent thermal resistances seen from the silicon to PCM (including the silicon layer, packaging layers, TIM, and the copper box) and from PCM to air, respectively. In Equation (2), the first term represents the power entering the PCM and the second term represents the power exiting the PCM. $E_{STORED,t}$ is the latent heat energy stored in the PCM at time t and $t_{sampling}$ is the sampling interval. We use $t_{sampling} = 1sec$, as the CPU temperatures are recorded at a rate of 1.0 Hz. Equation (3) is a simple accumulation operation, which approximates taking the integral of the net input power over time.

Equations (2) and (3) incur negligible computation overhead, since they consist of very few arithmetic operations. We measure this overhead in terms of CPU utilization on our testbed. We find that the PCM monitor overhead, including the temperature sensing and the calculations, is less than 0.4%, where 100% corresponds to fully utilizing all 4 cores.

We re-compute $R_{Si\_to\_PCM}$ and $R_{PCM\_to\_AIR}$ dynamically at runtime based on a model we construct for these thermal resistance values. Based on supporting experimental observations, we model $R_{Si\_to\_PCM}$ as proportional to the natural log of the difference in temperature between the CPU and PCM layer, as shown in Equation (4). We model $R_{PCM\_to\_AIR}$ similarly, in Equation (5).

$$R_{Si\_to\_PCM} = c_0 \cdot \ln(T_{CPU} - T_{PCM}) + c_1 \quad (4)$$

$$R_{PCM\_to\_AIR} = c_2 \cdot \ln(T_{PCM} - T_{AMB}) + c_3 \quad (5)$$

To derive the constants c_0, c_1, c_2, and c_3, we first record the CPU, PCM, ambient temperature, and CPU power dissipation while the system is idle. Using Equation (2), we solve for $R_{Si\_to\_PCM}$ and $R_{PCM\_to\_AIR}$ by setting P_{NET} to zero at steady state (since temperatures are settled and the system is dissipating a fixed amount of power). We assume that 50% of the heat generated by the CPU is dissipated to the PCM layer, while the rest is dissipated to the PCB. Therefore, the first term of Equation (2) is equal to 50% of the measured CPU power dissipation. We repeat this experiment at various CPU frequencies to obtain $R_{Si\_to\_PCM}$ and $R_{PCM\_to\_AIR}$ values that correspond to each frequency setting. We plot $R_{Si\_to\_PCM}$ as a function of the temperature difference between CPU and PCM, and use the least squares method to obtain a logarithmic regression line that best fits the plotted data points. This yields the constants $c_0 = 0.7698$ and $c_1 = 1.0726$ for Equation (4). We use the same method to determine $c_2 = -11.32$ and $c_3 = 71.81$, plotting $R_{PCM\_to\_AIR}$ as a function of the temperature difference between PCM and air.

Evaluation of Runtime Management Policies

In this section, we evaluate various runtime management policies. The first set of policies, which we denote as PCM-agnostic policies, take action based on core temperatures only. The second set of policies are PCM-aware management schemes, which monitor the remaining PCM capacity and take actions based on both PCM capacity and core temperatures. We describe the details of the policies below.

PCM-Agnostic Policies

Basic Sprinting. As our baseline, we implement the *computational sprinting* policy from prior work [1]. In this policy, during the *sprinting* mode, all four cores are activated until any of the cores reach the critical temperature (i.e., 80^oC for our case). Once the critical temperature is reached, the system switches to the *sustained* mode by migrating the threads to a single core and turning off the rest of the cores. The operation continues in *sustained* mode until the benchmark execution finishes.

Improved Sprinting. We implement this policy, which is not included in prior work, as an improved version of the *basic sprint-*

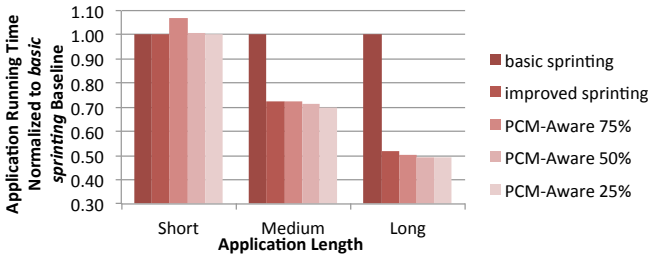


FIGURE 7: Benchmark running times for the policies and application lengths normalized to *basic sprinting*.

ing policy. In this policy, when the system switches to *sustained* mode, it does not stay in that mode until the benchmark finishes. Instead, the policy allows switching back to *sprinting* mode once the CPU temperature drops to a predefined value (i.e., 70^oC).

Temperature Triggered DVFS (tt-dvfs). This is a well known temperature triggered DVFS policy. This policy decreases the V/f level in steps if any of the cores reach the critical temperature (i.e., 80^oC). When the temperature falls to a predefined value (i.e., 70^oC), the V/f is increased back in steps. We add a feature to it such that it takes proactive action before hitting the temperature threshold. Thus, when any of the cores reach 75^oC *tt-dvfs* decreases the V/f level by N steps. We test cases for N = 1, 2 (i.e., N = 1 means switching from 1242 MHz to 1026 MHz, whereas N = 2 means switching from 1242 MHz to 810 MHz).

PCM-Aware Policy

A PCM-aware policy was proposed in prior work [2]. In this policy, when the remaining PCM latent heat capacity falls to 50%, the system switches to a *lower-intensity sprinting* mode by changing the V/f to a lower level. The aim is to use the PCM capacity at a lower rate, thus, extend the sprinting duration. In that sense, this policy also has a proactive nature. This PCM-aware policy uses a single PCM capacity threshold (i.e., 50%) mechanism. Depending on the application characteristics (i.e., application length, power consumption level, V/f level versus performance relationship), using different PCM capacity thresholds or different sprinting intensities may provide higher performance. We investigate this by experimenting with 3 parameters: (i) application length (i.e., short, medium, long), (ii) the number of steps N (i.e., N = 1, 2) to use when stepping down the V/f, (ii) threshold for the remaining PCM capacity (i.e., 25%, 50%, 75%) to analyze how they affect the resulting application performance.

RESULTS

Impact of Application Length. We first compare the *sprinting*, *improved sprinting*, and *PCM-aware* policies to show the impact of application length on the performance of each policy. For this purpose, we run the *sor* benchmark application with 3 different durations (short, medium, long). For the *PCM-aware* policy, we examine three cases with varying remaining PCM capacity thresholds: 25%, 50%, and 75%. For example, 25% threshold means that the policy switches to *lower-intensity sprinting* mode when the remaining PCM latent heat capacity falls to 25%. For all policies, we set the CPU frequency to 1242 MHz during *sprinting* mode. During the *lower-intensity sprinting* mode of the *PCM-aware* policy, we step down the CPU frequency by one step to 1026 MHz.

Figure 7 plots the benchmark application running time for each policy, as compared to the running time of the *basic sprinting* policy for each application duration. Smaller bars represent faster running times. For short applications, the *basic sprinting* performs the best, as it runs fast and finishes early without hitting the temperature threshold in that short amount of run time. On the other hand, *PCM-aware 75%* steps down the frequency and operates at a *lower-intensity sprinting* mode even though thermally it was not needed, which poses a performance penalty. For medium and long duration applications, the *basic sprinting* policy starts to perform poorly, since it consumes the PCM capacity to exhaustion fast and application is mostly executed with only one active core. *PCM-aware* policies offer better performance than the baseline policies by delaying temperature violation and thus, avoiding single core operation for most of the execution.

***tt-dvfs* versus *PCM-aware*.** Taking proactive actions before exhausting the thermal headroom provides performance benefits. This is because operating in high intensity modes uses up the PCM capacity sooner, after which the cores spend significant amount of time in the lowest V/f level. The next set of experiments investigate the two policies, where the proactive decisions are made based on core temperatures (i.e., when cores reach 75^oC for *tt-dvfs*) versus the remaining PCM capacity (i.e., *PCM-aware*). Figure 8 compares the running time resulting from both policies. From the Figure, we see that *PCM-aware 75%* gives 4.5% (N = 1) and 3.6% (N = 2) higher performance compared to *tt-dvfs*, as it takes action earlier than the *tt-dvfs*. Average CPU temperatures for the *tt-dvfs* and *PCM-aware 75%* cases are 75.4^oC and 73.5^oC, respectively. Moreover, with the *tt-dvfs* policy, cores spend 64% of the running time above 75^oC, compared to 20% when using the *PCM-aware 75%* policy.

One may adjust the 75^oC temperature setting for the *tt-dvfs* policy to match the behavior of *PCM-aware 75%*. However, for applications with different power characteristics, *PCM-aware* policies would take action at different temperatures as opposed to *tt-dvfs*. For example, for a low power application, having a 75% PCM capacity may correspond to a different CPU temperature compared to a high power application.

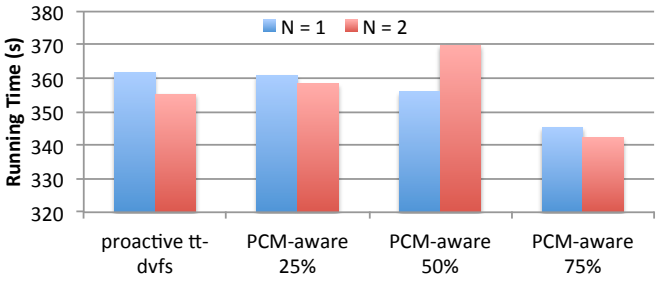


FIGURE 8: Running time for the *tt-dvfs* and *PCM-aware* policies.

CONCLUSION

In this paper, we have implemented a hardware testbed with a PCM container installed on top of the processor package. We experimentally validate our previously proposed phase change thermal model using the measurements on the testbed. Our transient simulations show that the error between the measured and simulated temperatures is less than $\mathbf{4^oC}$ with $\mathbf{0.63}$ probability. We then implement a soft PCM capacity sensor to monitor the remaining unmelted PCM at runtime. We evaluate a set of management policies and show an example use of the PCM sensor. Experimental evaluation shows that using *PCM-aware* policy with 75% capacity threshold gives up to **4.5**% higher performance compared to *tt-dvfs* policy.

REFERENCES

[1] Raghavan, A., Luo, Y., Chandawalla, A., Papaefthymiou, M., Pipe, K. P., Wenisch, T. F., and Martin, M. M. K., 2012. "Computational sprinting". In International Symposium on High Performance Computing (HPCA), pp. 1–12.

[2] Raghavan, A., Emurian, L., Shao, L., Papaefthymiou, M., Pipe, K. P., Wenisch, T. F., and Martin, M. M., 2013. "Computational sprinting on a hardware/software testbed". In International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), pp. 155–166.

[3] Tilli, A., Bartolini, A., Cacciari, M., and Benini, L., 2012. "Don't burn your mobile!: safe computational re-sprinting via model predictive control". In International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), pp. 373–382.

[4] Shao, L., Raghavan, A., Emurian, L., Papaefthymiou, M. C., Wenisch, T. F., Martin, M. M., and Pipe, K. P., 2014. "On-chip phase change heat sinks designed for computational sprinting". In Semiconductor Thermal Measurement and Management Symposium (SEMI-THERM), pp. 29–34.

[5] Kaplan, F., De Vivero, C., Howes, S., Arora, M., Homayoun, H., Burleson, W., Tullsen, D., and Coskun, A., 2014. "Modeling and analysis of phase change materials for efficient thermal management". In International Conference on Computer Design (ICCD), pp. 256–263.

[6] Skadron, K., Stan, M., Huang, W., Velusamy, S., Sankaranarayanan, K., and Tarjan, D., 2003. "Temperature-aware microarchitecture". In International Symposium on Computer Architecture (ISCA), pp. 2–13.

[7] Alawadhi, E., and Amon, C. H., 2003. "Pcm thermal control unit for portable electronic devices: experimental and numerical studies". *IEEE Transactions on Components and Packaging Technologies,* ***26***(1), March, pp. 116–125.

[8] Tan, F., and Fok, S. C., 2007. "Thermal management of mobile phone using phase change material". In Electronics Packaging Technology Conference, pp. 836–842.

[9] Yoo, D., and Joshi, Y., 2004. "Energy efficient thermal management of electronic components using solid-liquid phase change materials". *IEEE Transactions on Device and Materials Reliability,* ***4***(4), pp. 641–649.

[10] Stupar, A., Drofenik, U., and Kolar, J., 2010. "Application of phase change materials for low duty cycle high peak load power supplies". In International Conference on Integrated Power Electronics Systems (CIPS), pp. 1–11.

[11] Lingamneni, S., Asheghi, M., and Goodson, K., 2014. "A parametric study of microporous metal matrix-phase change material composite heat spreaders for transient thermal applications". In Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm), pp. 870–875.

[12] Pozo, R., and Miller, B. SciMark 2.0. On the WWW. URL `http://math.nist.gov/scimark2/index.html`.

[13] Ogoh, W., and Groulx, D., 2012. "Effects of the heat transfer fluid velocity on the storage characteristics of a cylindrical latent heat energy storage system: a numerical study". *Heat and Mass Transfer,* ***48***(3), pp. 439–449.

[14] Himran, S., Suwono, A., and Mansoori, G. A., 1994. "Characterization of Alkanes and Paraffin Waxes for Application as Phase Change Energy Storage Medium". *Energy Sources,* ***16***, pp. 117–128.

[15] Li, S., et al., 2009. "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures". In MICRO, pp. 469–480.

[16] Plummer, J. *Silicon VLSI Technology: Fundamentals, Practice, and Modeling*. Pearson Education.

[17] Kutz, M., 1998. *Mechanical Engineers' Handbook*. A Wiley-Interscience publication. Wiley.

[18] Narumanchi, S., Mihalic, M., Kelly, K., and Eesley, G., 2008. "Thermal interface materials for power electronics applications". In Thermal and Thermomechanical Phenomena in Electronic Systems, 2008. ITHERM 2008. 11th Intersociety Conference on, pp. 395–404.